Introduction

Support Vector Machines (SVM) remain one of the popular data classification tools in Machine Learning. In this work, we first look at optimizing the loss function (a minimization problem) in the primal form using (sub)gradient-descent. We next consider the dual form of the problem and optimize it (a maximization problem) via the projected gradient-ascent method, and a co-ordinate ascent technique, called box-constrained optimization.

Part 1: SVM: the primal form

Although many models exist to find a decision boundary to classify data, for example, the perceptron algorithm is guaranteed to find a decision boundary for linearly separable data. But the results depend highly upon the initialization. SVM is a more systematic and an optimal approach for finding the decision boundary - in a way that maximizes the buffer around. The goal of SVM is to find that hyperplane that has the maximum margin m from the closest points. In other words, we maximize the minimum distance, i.e. a mini-max problem. A bit of mathematical manipulation gives us the following objective function for linearly separable data $(x^1, y^1), (x^2, y^2), ..., (x^n, y^n)$:

$$\min_a cr(a)$$

$$\text{s.t. } a^T x^{(i)} y^{(i)} \geq m \; \forall 1 \leq i \leq n$$

where m is the minimum expected distance between the data-points and the separating hyperplane, and $r(a)$ is a regularization term (here L2).

If the classes are not linearly separable, we introduce slack variables ($\epsilon \geq 0 \in \mathbf{R}^n$) for each data-point that allow margin violation by some value, which we want to minimize. Thus the objective becomes[2]:

$$\min_{a,\epsilon} cr(a) + \sum_{i=1}^{n} \epsilon_i$$

$$\text{s.t. } a^T x^{(i)} y^{(i)} \geq m - \epsilon_i \; \forall 1 \leq i \leq n$$

The constraints can be rewritten as:

$$a^T x^{(i)} y^{(i)} \geq m - \epsilon_i$$
$$a^T x^{(i)} y^{(i)} - m \geq -\epsilon_i$$
$$m - a^T x^{(i)} y^{(i)} \leq \epsilon_i$$

Now since $\epsilon \geq 0$, $\implies m - a^T x^{(i)} y^{(i)} \leq \epsilon_i = \max(0, m - a^T x^{(i)} y^{(i)})$ and since we want to minimize ϵ_i, then the minimum value that ϵ_i can assume is either 0 or $m - a^T x^{(i)} y^{(i)}$ but not greater. Then the constraint becomes:

$$\max(0, m - a^T x^{(i)} y^{(i)}) = \epsilon_i$$

Therefore substituting this definition of (ϵ_i), the objective can be written as an unconstrained problem:

$$\min_a \sum_{i=1}^{n} \max(0, m - a^T x^{(i)} y^{(i)}) + cr(a)$$

Understanding the objective function

With a choice of $m = 1$, $\forall i$ for which the classifications are correct, $a^T x^{(i)} y^{(i)} \geq 0$ and

$$m - a^T x^{(i)} y^{(i)} \begin{cases} 0 \leq m - a^T x^{(i)} y^{(i)} \leq 1, \text{if } m > a^T x^{(i)} y^{(i)} \\ m - a^T x^{(i)} y^{(i)} < 0, \text{if } m < a^T x^{(i)} y^{(i)} \end{cases}$$

In other words, $\max(0, m - a^T x^{(i)} y^{(i)}) \in (-\infty, 1] \implies$ small values of the objective function.

$\forall i$ for which the classifications are incorrect, $a^T x^{(i)} y^{(i)} \leq 0$ and

$$m - (-a^T x^{(i)} y^{(i)}) = m + a^T x^{(i)} y^{(i)}$$

In other words, $\max(0, m - a^T x^{(i)} y^{(i)}) \in [1, \infty) \implies$ large values of the objective function.

Figure 1 gives a glimpse of how the objective loss function we defined looks like in practical settings (the hinge-loss).

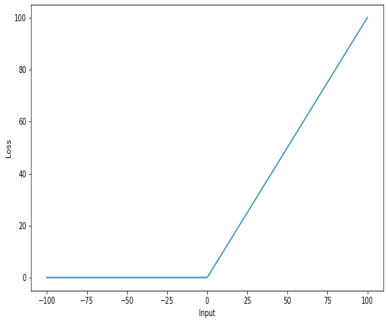

Figure 1: Hinge loss - objective function of primal form of SVM

Sub-gradient descent on the primal problem

Now we consider minimizing the objective function thus defined. Although hinge-loss function is convex, it is not differentiable at all points as can be seen from Figure 1. But sub-gradients exist for the whole domain of the function and,

$$a^{(t+1)} = a^t - \eta g, \; g \in \delta f(a^t)$$

where g represents the sub-gradient and $\delta f(a^t)$ the sub-differential. In our case,

$$g \in \delta f(a^t) \begin{cases} -x^{(i)} y^{(i)} + ca, \text{if } a^T x^{(i)} y^{(i)} > 0 \\ (0, x^{(i)} y^{(i)} + ca), \text{if } a^T x^{(i)} y^{(i)} = 0 \\ ca, \text{if } a^T x^{(i)} y^{(i)} < 0 \end{cases}$$

where c represents the regularization weight. We make an arbitrary choice to fix the range of g at the hinge to be $x^{(i)} y^{(i)}$ and thus we get,

$$g \in \delta f(a^t) \begin{cases} -x^{(i)} y^{(i)} + ca, \text{if } a^T x^{(i)} y^{(i)} \geq 0 \\ ca, \text{if } a^T x^{(i)} y^{(i)} < 0 \end{cases}$$

Results

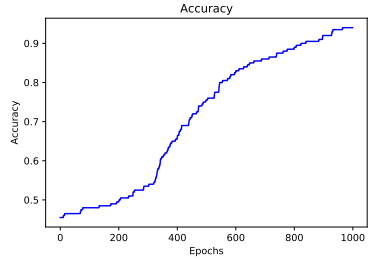

Figure 2: Accuracy

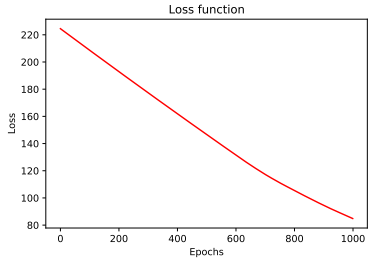

Figure 3: Loss function

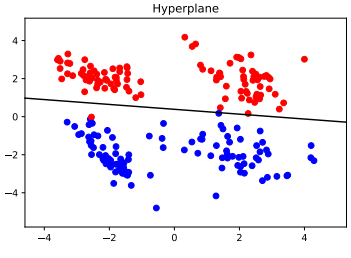

Figure 4: Hyperplane

Pseudocode

Algorithm 1: Sub-gradient descent on the primal problem

Input: Learning rate: α, Regularization weight: c, Number of epochs: n_epochs
Output: Weights of the hyperplane: a [n_features, 1]
Data: Features: X [n_samples, n_features], Labels: y [n_samples, 1]

```
a := initializeWeights(X.shape[1])
for 0...n_epochs do
    /* Compute cost                                                    */
    obj := X@a*y
    obj := 1 - obj
    /* Get values from which the cost is greater or equal than 0       */
    mask := obj ≥ 0
    /* Select only those values from which the cost is greater or equal than 0 */
    y_aux := y*mask
    /* Compute subgradient                                             */
    subgradient = (-y_aux.T@X + c*a)*α
    /* Update weights a                                                */
    a := a - subgradient
return a
```

Part 2: Projected gradient-ascent

The primal objective function might be difficult to solve. The function is possibly non-convex and the subgradient may not be a descent direction. Besides, the subgradient allows to move closer to the optimal solution in terms of euclidean distance, but it is difficult to check if an iteration improves the distance to the optimal point since it is obviously unknown. The concept of Duality gives us an alternative form of the primal, which is guaranteed to be concave, giving us an easy maximization problem. If strong duality holds, the solution to the dual problem holds for the primal problem too. If strong duality doesn't hold (i.e. weak duality), we still get a lower bound on the objective function value. An attempt is made to explain all this in the following sections.

An intuition of the Lagrangian formulation

Typical optimization problems are as follows:

$$\min_a f(x)$$

$$\text{s.t. } g_i(x) \leq 0 \ \forall i$$

One way to tackle this problem is to write in an equivalent unconstrained form, such that the there is an infinite penalization if the constraints are violated[3]. We can do this by adding a function of the constraints $_i(x)$ to $f(x)$:

$$f(x) + I(g_i(x)) \begin{cases} f(x), \text{if } g_i(x) \leq 0 \text{i.e. constraints respected} \\ \infty, \text{otherwise i.e. constraints violated} \end{cases}$$

This formulation gives an equivalence to the original objective, but it is discontinuous step function. We can instead take some linear factor $\lambda_i \geq 0$,

$$f(x) + \sum_i \lambda_i g_i(x) = L(\lambda, x)$$

This is the Lagrangian, an affine function (thus differentiable) and maximizing it w.r.t. λ give us our original objective, only unconstrained.

$$\max_\lambda (f(x) + \sum_i \lambda_i g_i(x)) \begin{cases} f(x), \text{if } g_i(x) \leq 0 \implies \lambda \lim \infty \text{ i.e. constraints respected} \\ \infty, \text{if } g_i(x) > 0 \implies \lambda \lim 0 \text{ i.e. constraints violated} \end{cases}$$

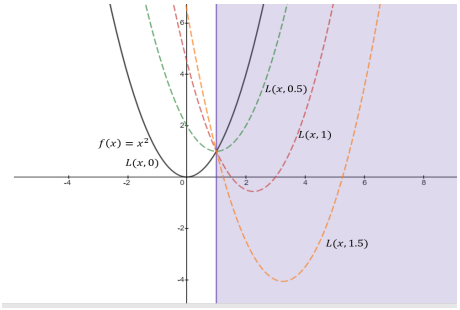

Figure 5: Lagrangian functions visualized for $f(x) = x^2, g(x) >= 1$

A natural Dual formulation

Now we have an unconstrained equivalent form of our function with constraints, which we want to minimize w.r.t x, we have:

$$\min_x \max_\lambda L(\lambda, x)$$

With a change in order, we get:

$$\max_\lambda \min_x L(\lambda, x)$$
$$= \max_\lambda g(\lambda)$$

where $g(\lambda) = \min_x L(\lambda, x)$. As mentioned before, since $L(\lambda, x)$ is an affine function, it is known that pointwise minimization of an affine function is concave. Our original objective thus has an equivalent concave maximization problem (see Figure 2).

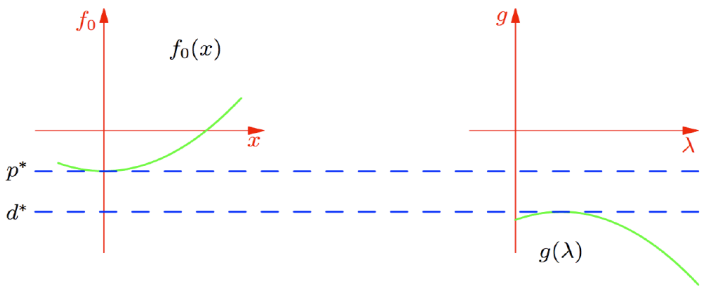

Figure 6: Weak duality[5]

Dual form of the SVM primal problem

From the discussion of the precious section, we have a L2 regularized L1 Support Vector Classification loss function:

$$\min_a \sum_{i=1}^{n} \max(0, m - a^T x^{(i)} y^{(i)}) + cr(a)$$

By introducing an equality constraint, the primal problem can be rewritten as,

$$\min_{a,v} \sum_{i=1}^{n} \max(0, 1 - v_i) + cr(a)$$

$$\text{s.t. } v_i = a^T x^{(i)} y^{(i)}$$

Introducing the Lagrangian,

$$L(a, v_i, \lambda_i) = \sum_{i=1}^{n} \max(0, 1 - v_i) + cr(a) + \lambda_i (a^T x^{(i)} y^{(i)} - v)$$

Let $l(v_i) = \max(0, 1 - v_i)$ represent the hinge loss. Then the dual form is (see[2]),

$$= \max_{\lambda_i} \min_v \sum_{i=1}^{n} (l(v_i) - \lambda_i v_i) + \min_a (r(a) + \lambda_i a^T x^{(i)} y^{(i)})$$

$$\max_{\lambda_i} \sum_{i=1}^{n} \min_{v} \left(l(v_i) - \lambda_i v_i\right) - \max_{a}(-\lambda_i a^T x^{(i)} y^{(i)} - r(a))$$

$$\max_{\lambda_i} - \sum_{i=1}^{n} \max_{v} \left(\lambda_i v_i - l(v_i)\right) - \max_{a} a^T(-X^T Y\lambda) - \frac{c}{2}||a||^2$$

$$= \max_{\lambda} \ - \sum_{i=1}^{n} l^*(\lambda_i) - cr^*(-X^T Y\lambda)$$

$$= \max_{\lambda} - \sum_{i=1}^{n} \lambda_i - \frac{c}{2}\lambda^T Y X X^T Y \lambda$$

$$\forall 1 \leq i \leq n: \ -1 \leq \lambda_i \leq 0$$

Again, we note that,

- the objective is concave and differentiable
- dual variables λ are constrained to be in a convex set (linear constraints)

Intuition behind Fenchel Conjugate

The counterpart of the Fourier transform in Convex Analysis is the Fenchel conjugate [3]. It allows us to simplify the dual of the a problem.

$$f^*(a) = \max_{x} \left(a^T x - f(x)\right)$$

The definition can be interpreted as that $f^*(a)$ is simply the maximum amount that the linear functional $a^T x$, exceeds $f(x)$.

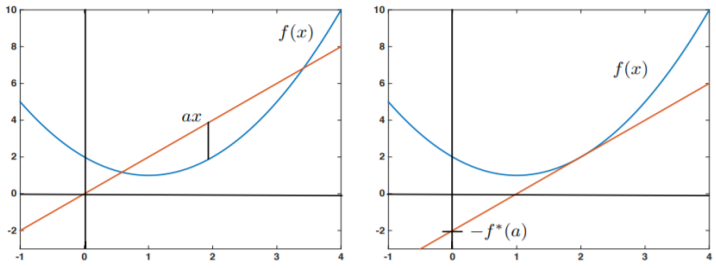

Figure 7: Fenchel conjugate[4]

It shows that a function f can be represented as a supremum of affine functions $x \longrightarrow \langle a, x\rangle - f^*(a)$, where $f^*(a)$ determines the constant term of the affine function with slope a[1].
The conjugate does not give us anything new itself. But it enables us to derive the dual problems more quickly. For example, for hingle loss:

$$l^*(\lambda_i) = \begin{cases} \lambda_i, \text{ if } \lambda_i \in [-1, 0] \\ \infty, \text{ otherwise} \end{cases}$$

In the derivation of dual form of the SVM primal problem, we are able to manipulate the minimization terms in a form of maximum over affine functions, which becomes equivalent to the conjugate form as discussed, simplifying the overall optimization problem from a max-min, to a maximization problem of a concave function (since $f^*(a)$ is the point-wise maximum of affine functions in a "indexed" by x, $f^*(a)$ is always concave).

Therefore, whenever we have some terms in our problem in the form of $\max_{x} \left(a^T x - f(x)\right)$ or even $\min_{x} \left(a^T x - f(x)\right)$, we should be replace the term with the Fenchel conjugate which is much easier to deal with.

Projected gradient ascent

We can now look at optimizing, i.e. finding the maxima of the dual. Although Gradient Ascent is the goto method, for problems with constraints such as ours (the dual variables $\lambda_i \in [-1, 0]$), we can tune the Gradient Ascent approach to take into account constraints via projection of the constrained variables into their valid domain. Therefore, Projected gradient ascent has just one more step than gradient ascent, the projection step:

$$\max_{\lambda} -\sum_{i=1}^{n} \lambda_i - \frac{c}{2}\lambda^T Y X X^T Y \lambda$$

$$\forall 1 \leq i \leq n: \quad -1 \leq \lambda_i \leq 0$$

Let $f(\lambda) = -\sum_{i=1}^{n} \lambda_i - \frac{c}{2}\lambda^T Y X X^T Y \lambda$. Then for each simulation step,

$$\lambda^{(t+1)} = \underset{[-1,0]}{\text{Proj}}[\lambda^t + \eta \nabla f(\lambda^t)]$$

where $\nabla f(\lambda^t) = -1 - cYXX^TY\lambda^t$

We can go from the dual variable $/lambda$ space to the original parameter space a via,

$$\nabla L(a, v_i, \lambda) = 0$$

$$\implies \nabla(\sum_{i=1}^{n} \max(0, 1 - v_i) + cr(a) + \lambda_i(a^T x^{(i)} y^{(i)} - v)) = 0$$

$$\implies ca + (\lambda^T Y X)^T = 0 \implies a = -\frac{1}{c} X^T Y \lambda$$

Results

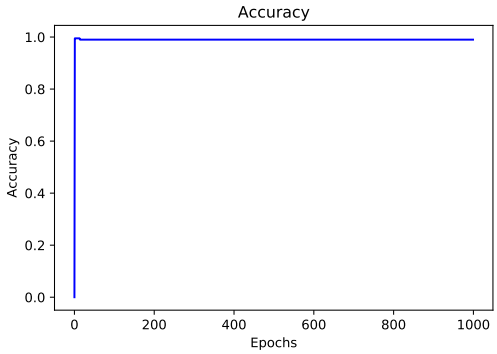

Figure 8: Accuracy

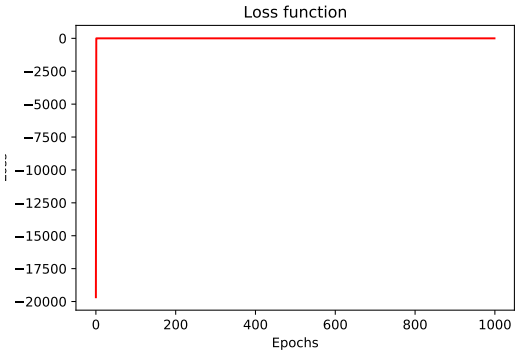

Figure 9: Loss function

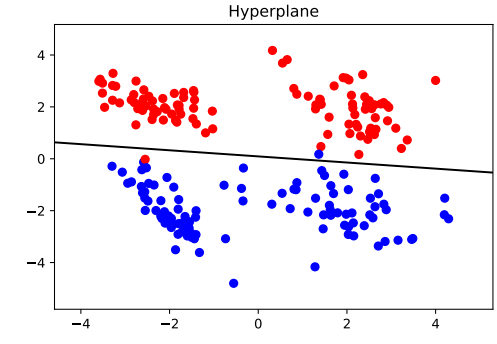

Figure 10: Hyperplane

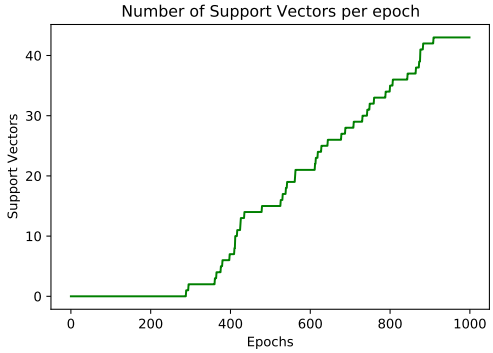

Figure 11: Support vectors

Pseudocode

Algorithm 2: Projected gradient ascent on the dual problem

Input: Learning rate: α, Regularization weight: c, Number of epochs: n_epochs
Output: Weights of the hyperplane: a [n_features, 1]
Data: Features: X [n_samples, n_features], Labels: y [n_samples, 1]

```
dual_vars := initializeDualVars(X.shape[0])
Y := diagonalize(y)                          // Create a diagonal matrix with the values of y
for 0...n_epochs do
    gradient := - 1 - c * (Y @ X @ X.T @ Y @ dual_vars)        // Shape: [n_examples, 1]
    dual_vars := dual_vars + α * gradient
    dual_vars = clip(dual_vars, a_min=-1, a_max=0)
a := - (X.T @ Y @ dual_vars) / c
return a
```

Part 3: Box-constrained optimization

In co-ordinate optimization methods, at each iteration, the (dual) objective is optimized with respect to a single (dual) variable, while the rest of the (dual) variables are kept in tact (can replace individual coordinates with blocks of coordinates). An advantage this brings is cheaper iterations. It feels intuitve that instead of folowing the direction of greatest descent (gradient) looping over all directions (co-ordinates) and then minimizing the objective must be wasteful. But "in most regimes" authors in[6] conclude that Stochastic Dual Co-ordinate Ascent (SDCA) converges faster than Stochastic Gradient Descent (SGD). Also, there is no step-size hyperparameter and it is the most used for large-scale LASSO and SVM optimization problems.

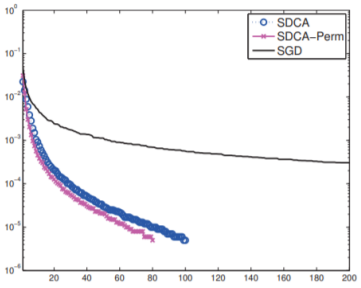

Figure 12: Stochastic co-ordinate descent vs stochastic gradient descent for SVM dual problem[6]

The SVM dual problem can be written as,

$$\max_{\lambda} \frac{1}{2}\lambda^T Q\lambda + b^T\lambda$$

$$\textbf{s.t.} \quad -1 \leq \lambda \leq 0$$

where $Q = -cYXX^TY$ (symmetric by definition) and $b = -1$ Setting the partial derivative w.r.t each co-ordinate (λ_k) equal to zero gives,

$$\lambda_k = \frac{-b_k - \sum_{i\neq k} \lambda_i Q_{k,i}}{Q_{k,k}}$$

To ensure the constraints on λ are respected:

$$\lambda_k = \underset{[-1,0]}{\text{Proj}}\left[\frac{-b_k - \sum_{i\neq k} \lambda_i Q_{k,i}}{Q_{k,k}}\right]$$

Results

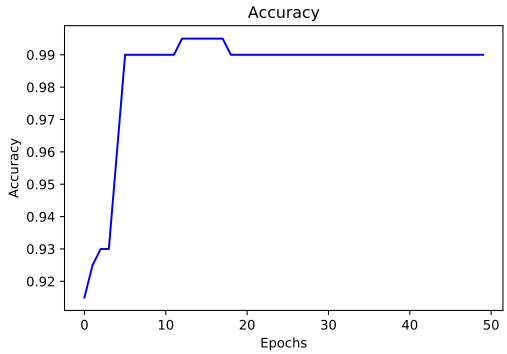

Figure 13: Accuracy

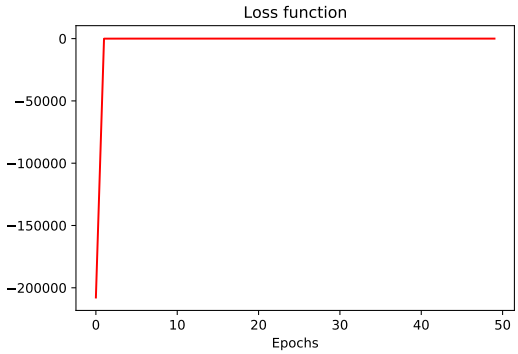

Figure 14: Loss function

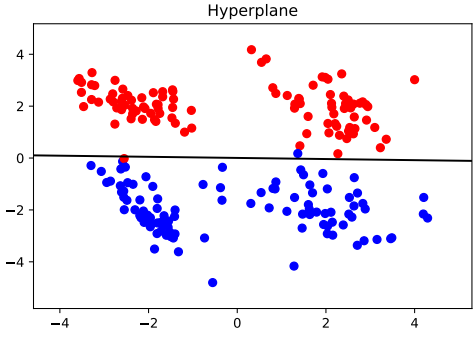

Figure 15: Hyperplane

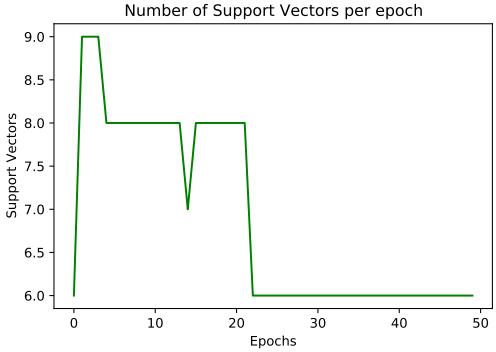

Figure 16: Support vectors

Pseudocode

Algorithm 3: Example code

Input: Regularization weight: c, Number of epochs: n_epochs
Output: Weights of the hyperplane: a [n_features, 1]
Data: Features: X [n_samples, n_features], Labels: y [n_samples, 1]

```
dual_vars := initializeDualVars(X.shape[0])
Q := c * (- Y @ X @ X.T @ Y )
b := -1
Y := diagonalize(y)                          // Create a diagonal matrix with the values of y
for i=0...n_epochs do
    for k=0...n_features do
        mask = [0 if k == i else 1]
        dual_vars_k = (-b[k] - sum(dual_vars * Q[k] * mask) )/Q[k][k]
        dual_vars[k] = clip(dual_vars_k, a_min=-1, a_max=0)
a := - (X.T @ Y @ dual_vars) / c
return a
```

Conclusion

We saw the primal and dual formulation for Support Vector Machines. We attempted explanations behind the theory of the Lagrangian formulation, the dual, and the Fenchel Conjugates which are usually difficult to grasp intuitively. We then moved onto use a simple artificial dataset to put the theory into practice, by using Sub-gradient descent to solve the primal objective, and Gradient-ascent and Box-constrained optimization to to get the optima with the dual variables, from which we derive the parameters in the primal space.

References

[1] Heinz H. Bauschke and Yves Lucet. *What is a Fenchel conjugate?*

[2] Caio Corro. Class notes for tc1, university paris-sud. 2021.

[3] David Knowles. *Lagrangian Duality for Dummies*.

[4] J. Romber. A second look at duality.

[5] David S. Rosenberg. *Lagrangian Duality in 10 Minutes*.

[6] Shai Shalev-Shwartz and Tong Zhang. *Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization*, pages 567–599. Journal of Machine Learning Research 14 (2013) 567-599, 2013.